Assignment - III

Mohammad Wasiq

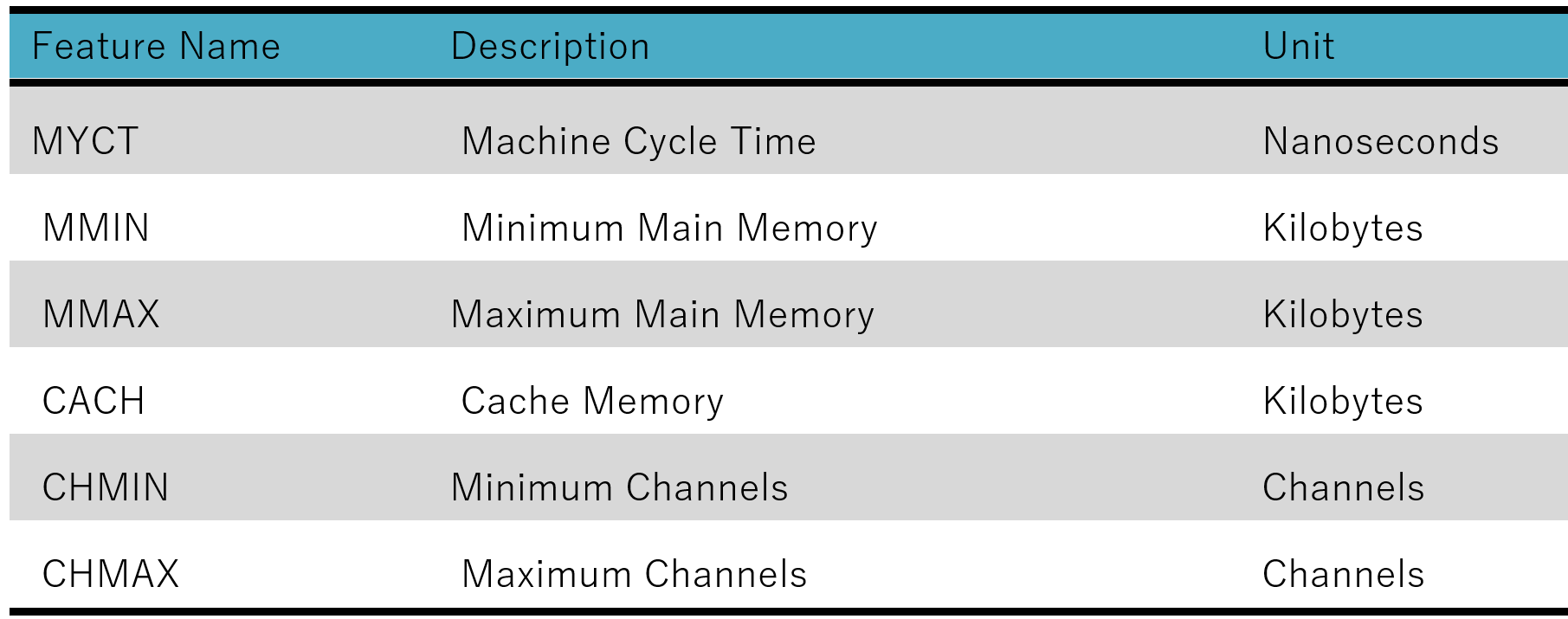
2022-06-22

Name -: **Mohammad Wasiq**

E-mail -: [**mohammadwasiq0786@gmail.com**](mailto:mohammadwasiq0786@gmail.com)

# First Problem

The data on features given in the table below are collected to estimate the published relative performance (PRP) of a centralised processing unit. The data is given in the CPU\_Data file.



**Load the Data**

df<- readxl::read\_excel("CPU\_data.xlsx")  
  
# Head of Data   
attach(df)  
  
# Head of Data   
head(df)

## # A tibble: 6 × 7  
## MYCT MMIN MMAX CACH CHMIN CHMAX PRP  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 125 256 6000 256 16 128 198  
## 2 29 8000 32000 32 8 32 269  
## 3 29 8000 32000 32 8 32 220  
## 4 29 8000 32000 32 8 32 172  
## 5 29 8000 16000 32 8 16 132  
## 6 26 8000 32000 64 8 32 318

1. Split the data randomly into training (80%) and test (20%). Develop a CART model for PRP using training data

library(rpart)  
library(caret)

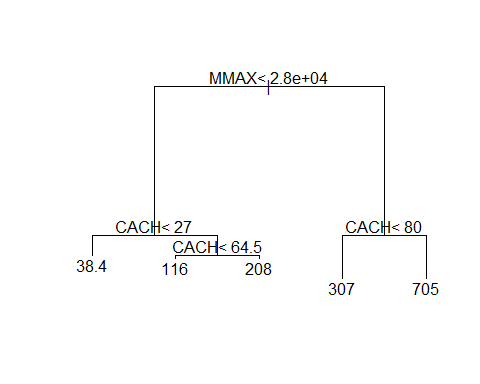
## Loading required package: ggplot2

## Loading required package: lattice

set.seed(1)  
indexes = createDataPartition(PRP, p = 0.80, list = F)  
train = df[indexes, ]  
test = df[-indexes, ]  
  
train\_x = train[, -7]  
train\_y = train[, 7] # PRP  
  
test\_x = test[, -7]  
test\_y = test[, 7] # PRP  
  
dim(train\_x)

## [1] 169 6

fit = rpart(PRP ~ ., data = train)  
  
  
par(xpd = NA) # otherwise on some devices the text is clipped  
plot(fit)  
text(fit, digits = 3)

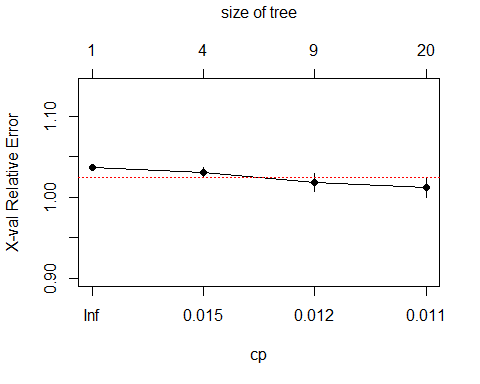


print(fit, digits = 2)

## n= 169   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 169 5000000 110   
## 2) MMAX< 2.8e+04 147 500000 61   
## 4) CACH< 27 113 58000 38 \*  
## 5) CACH>=27 34 190000 140   
## 10) CACH< 64 26 76000 120 \*  
## 11) CACH>=64 8 59000 210 \*  
## 3) MMAX>=2.8e+04 22 1800000 430   
## 6) CACH< 80 15 290000 310 \*  
## 7) CACH>=80 7 760000 700 \*

1. Show the cp plot and give the optimum cp value

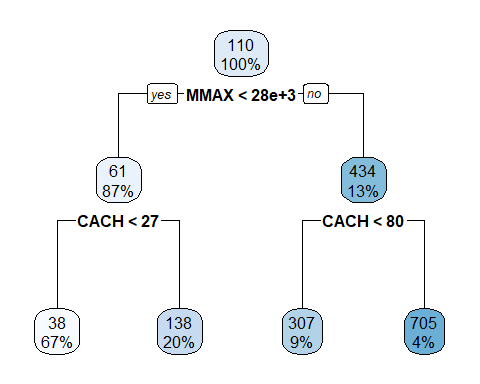
library(rpart.plot)  
mymodel = rpart(PRP ~. ,data = train, method = 'class', control = rpart.control(minsplit = 2))  
  
# Cross validation and identification of cp  
plotcp(mymodel, pch = 19, col = "red")



Optimum cp = Corresponding to minimum cross validation relative error

1. Display the best CART model obtained (rpart.plot) and give your interpretation

library(rpart.plot)  
fit.pruned = prune(fit, cp = 0.019)  
  
rpart.plot(fit.pruned)



1. Compute the mean square error (MSE) and root mean square error (RMSE) for training data. Is the model reasonably accurate?

pred\_y\_tr = predict(fit.pruned, train\_x)  
#Accuracy checking  
  
#Next, we'll check the prediction accuracy with MSE, MAE, and RMSE metrics.  
  
print(head(data.frame(train\_y, pred\_y\_tr)))

## PRP pred\_y\_tr  
## 1 198 137.9118  
## 2 269 307.4000  
## 3 220 307.4000  
## 4 132 137.9118  
## 5 318 307.4000  
## 6 367 307.4000

msetr = sapply((train\_y - pred\_y\_tr)^2, mean, 2)  
  
maetr = sapply(as.data.frame(train\_y, pred\_y\_tr), caret::MAE, 2)  
  
rmsetr = sapply(as.data.frame(train\_y, pred\_y\_tr), caret::RMSE, 2)  
  
tr\_acc <- cat("MSE: ", msetr, "MAE: ", maetr, " RMSE: ", rmsetr)

## MSE: 458.2631 MAE: 107.8994 RMSE: 202.5156

1. Validate the model on test data. Compute MSE and RMSE on test data

pred\_y = predict(fit.pruned, test\_x)  
#Accuracy checking  
  
#Next, we'll check the prediction accuracy with MSE, MAE, and RMSE metrics.  
  
print(head(data.frame(test\_y, pred\_y)))

## PRP pred\_y  
## 1 172 307.40000  
## 2 40 38.40708  
## 3 28 38.40708  
## 4 31 38.40708  
## 5 69 38.40708  
## 6 33 38.40708

mse = sapply((test\_y - pred\_y)^2, mean, 2)  
  
mae = sapply(as.data.frame(test\_y, pred\_y), caret::MAE, 2)  
  
rmse = sapply(as.data.frame(test\_y, pred\_y), caret::RMSE, 2)  
  
test\_acc<- cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)

## MSE: 529.3516 MAE: 85.55 RMSE: 131.649

1. Provide the comparison table of MSE & RMSE for training and test data. Give your comments on the model accuracy and generalizability?

data.frame(Errors = c("MSE", "MAE", "RMSE"),   
 Training\_accuracy = c(458.26, 107.89, 202.52),   
 Test\_accuracy = c(529.35, 85.55, 131.6)  
 )

## Errors Training\_accuracy Test\_accuracy  
## 1 MSE 458.26 529.35  
## 2 MAE 107.89 85.55  
## 3 RMSE 202.52 131.60

1. Validate the model on test data? Compute mean square error and root mean square on test data. Give your comments on model generalizability.
2. Develop a model to predict PRP using the Bagging method.

library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

mymodel = randomForest(PRP ~., data = train, mtry = 13, importance = TRUE)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid  
## range

mymodel

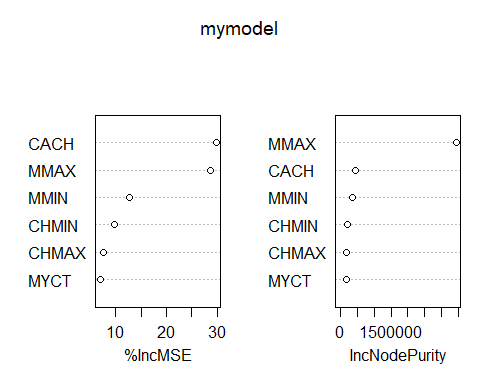
##   
## Call:  
## randomForest(formula = PRP ~ ., data = train, mtry = 13, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## Mean of squared residuals: 3468.082  
## % Var explained: 88.19

1. Provide variable importance plot and give your comments?

importance(mymodel)

## %IncMSE IncNodePurity  
## MYCT 7.000670 162707.3  
## MMIN 12.671986 358981.4  
## MMAX 28.691670 3433065.4  
## CACH 29.818329 446125.0  
## CHMIN 9.734622 214210.2  
## CHMAX 7.480474 184521.4

varImpPlot(mymodel)



1. Compute the R2, mean square error and root mean square on training data. Give your comments on model accuracy.

predtrain = predict(mymodel, newdata = train)  
restrain = train$PRP - predtrain  
mset = mean(restrain^2)  
  
R2t<- R2(predtrain, train$PRP, form = "traditional")  
  
rmset = sqrt(mse)  
cat("MSE: ", mset," RMSE: ", rmset, "R2: ", R2t)

## MSE: 988.0823 RMSE: 23.00764 R2: 0.9663577

1. Validate the model on test data? Compute mean square error and root mean square on test data. Give your comments on model generalizability.

predtest = predict(mymodel, newdata = test)  
restest = test$PRP - predtest  
mse = mean(restest^2)  
  
R2<- R2(predtest, test$PRP, form = "traditional")  
  
rmse = sqrt(mse)  
cat("MSE: ", mse," RMSE: ", rmse, "R2: ", R2)

## MSE: 1352.706 RMSE: 36.77915 R2: 0.8649003

1. Develop a model to predict PRP using the Random Forest method.

mymodel = randomForest(PRP ~., data = train, importance = TRUE)  
mymodel

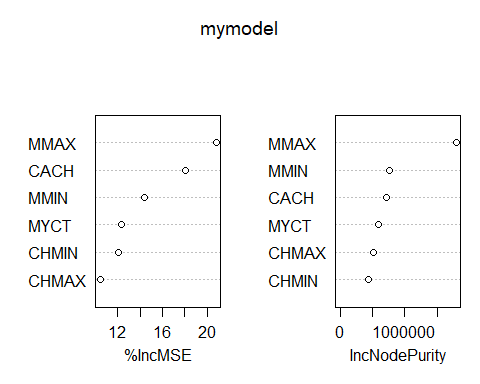
##   
## Call:  
## randomForest(formula = PRP ~ ., data = train, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## Mean of squared residuals: 3375.5  
## % Var explained: 88.51

1. Provide variable importance plot and give your comments?

importance(mymodel)

## %IncMSE IncNodePurity  
## MYCT 12.29317 593191.2  
## MMIN 14.38355 765801.3  
## MMAX 20.80375 1795686.3  
## CACH 18.07221 707943.6  
## CHMIN 12.02622 430274.9  
## CHMAX 10.46882 518347.5

varImpPlot(mymodel)



1. Compute the R2, mean square error and root mean square on training data. Give your comments on model accuracy.

predtrain = predict(mymodel, newdata = train)  
restrain = train$PRP - predtrain  
mset = mean(restrain^2)  
  
R2t<- caret::R2(predtrain, train$PRP, form = "traditional")  
  
rmset = sqrt(mse)  
cat("MSE: ", mset," RMSE: ", rmset, "R2: ", R2t)

## MSE: 989.406 RMSE: 36.77915 R2: 0.9663127

1. Validate the model on test data? Compute mean square error and root mean square on test data. Give your comments on model generalizability.

predtest = predict(mymodel, newdata = test)  
restest = test$PRP - predtest  
mse = mean(restest^2)  
  
R2<- R2(predtest, test$PRP, form = "traditional")  
  
rmse = sqrt(mse)  
cat("MSE: ", mse," RMSE: ", rmse, "R2: ", R2)

## MSE: 923.7535 RMSE: 30.39331 R2: 0.9077413

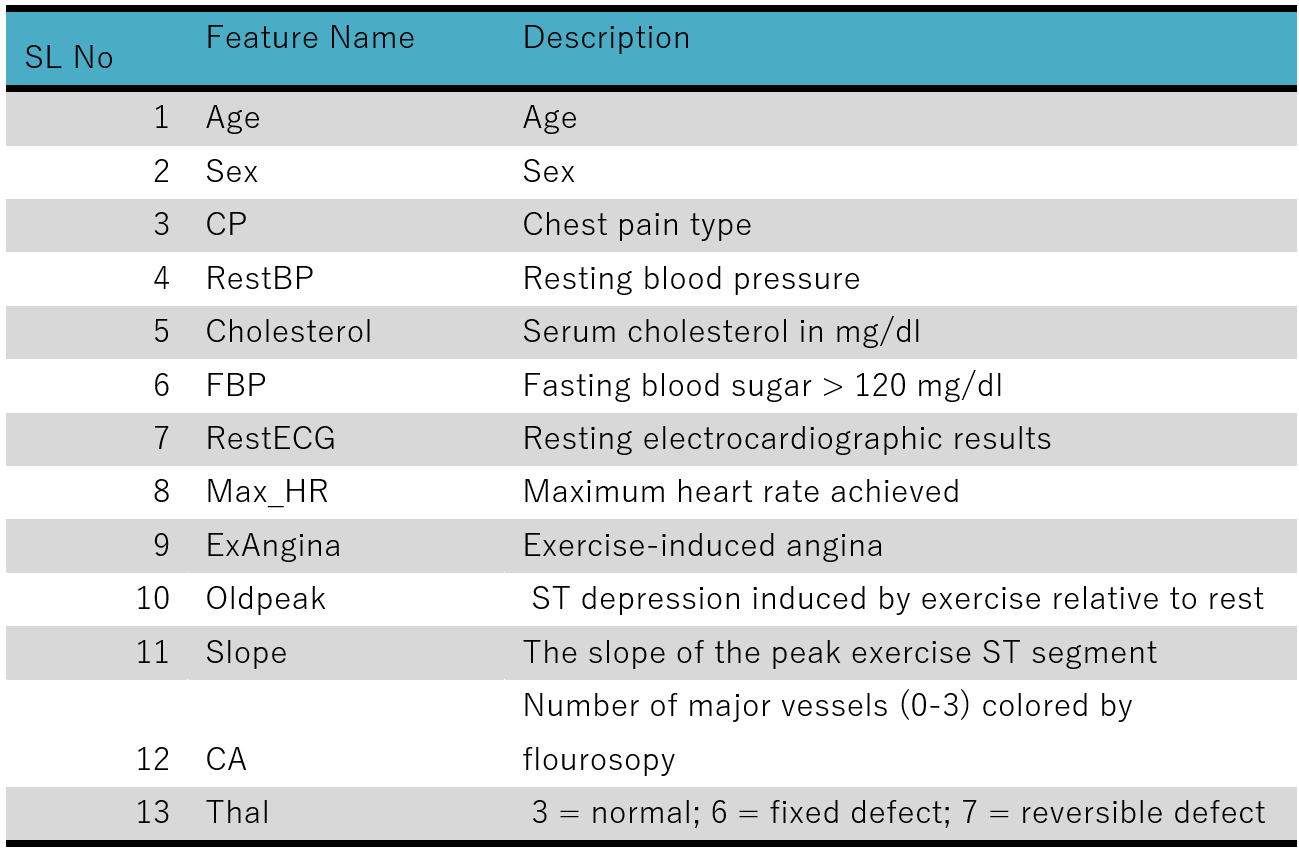
1. Compare the Regression tree, bgging & random forest models and give your comments.

From Regression tree, bgging & random forest models we can easily compute that the model using **Random Forest** is best most model as comparision because the value of **RMSE** of Random Forest Model is **36** and **30** for training and testing data respectively.

1. Compare the Regression tree, bgging & random forest models with the linear regression model of assignment 2 and give your comments.

From Regression tree, bgging & random forest models we can easily compute that the model using **Random Forest** is best most model as comparision because the value of **RMSE** of Random Forest Model is **36** and **30** for training and testing data respectively.

# Second Problem



**Load the Data**

df<- readxl::read\_excel("Heart\_Disease\_Data.xlsx")  
  
# Head of Data   
attach(df)  
  
# Head of Data   
head(df)

## # A tibble: 6 × 14  
## Age Sex CP RestBP Cholesteral FBP RestECG Max\_HR ExAngina Oldpeak  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 63 1 3 145 233 1 0 150 0 2.3  
## 2 37 1 2 130 250 0 1 187 0 3.5  
## 3 41 0 1 130 204 0 0 172 0 1.4  
## 4 56 1 1 120 236 0 1 178 0 0.8  
## 5 57 0 0 120 354 0 1 163 1 0.6  
## 6 57 1 0 140 192 0 1 148 0 0.4  
## # … with 4 more variables: Slope <dbl>, CA <dbl>, Thal <dbl>, Result <dbl>

dim(df)

## [1] 303 14

names(df)

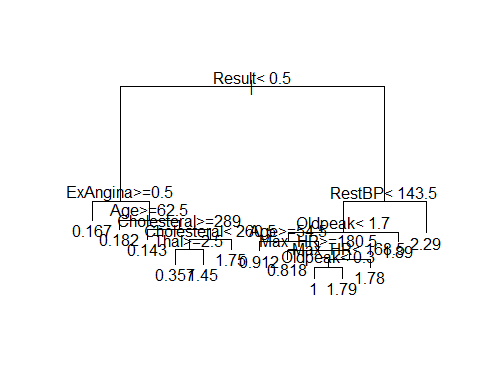
## [1] "Age" "Sex" "CP" "RestBP" "Cholesteral"  
## [6] "FBP" "RestECG" "Max\_HR" "ExAngina" "Oldpeak"   
## [11] "Slope" "CA" "Thal" "Result"

1. Split the data randomly into training (80%) and test (20%). Develop a classification tree model for Result

set.seed(1)  
indexes = createDataPartition(CP, p = 0.80, list = F)  
train = df[indexes, ]  
test = df[-indexes, ]  
  
train\_x = train[, -14]  
train\_y = train[, 14] # PRP  
  
test\_x = test[, -14]  
test\_y = test[, 14] # PRP  
  
dim(train\_x)

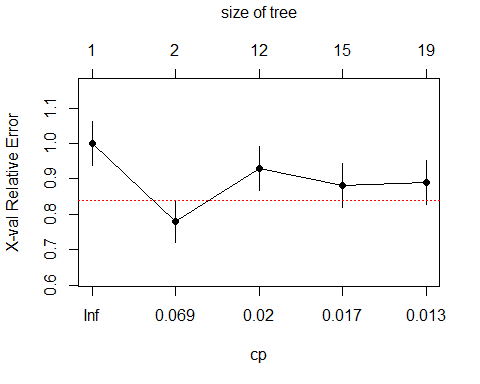
## [1] 244 13

fit = rpart(CP ~ ., data = train)  
  
  
par(xpd = NA) # otherwise on some devices the text is clipped  
plot(fit)  
text(fit, digits = 3)



1. Show the cp plot and give the optimum cp value

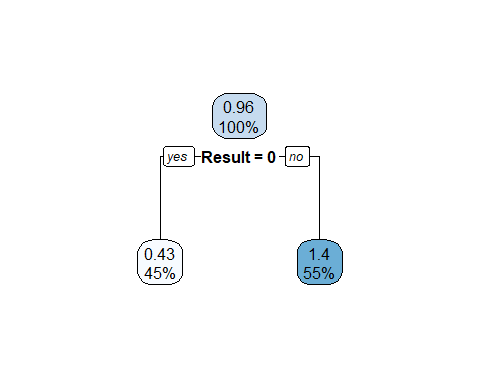
library(rpart.plot)  
mymodel = rpart(CP ~. ,data = train, method = 'class', control = rpart.control(minsplit = 2))  
  
# Cross validation and identification of cp  
plotcp(mymodel, pch = 19, col = "red")



Optimum Corresponding to minimum cross validation relative error

1. Display the best CART model obtained (rpart.plot) and give your interpretation

library(rpart.plot)  
fit.pruned = prune(fit, cp = 0.069)  
  
rpart.plot(fit.pruned)



1. Compute the actual versus predicted table, accuracy% and misclassification % on training data. Give your comments on model accuracy.

library("e1071")  
model<- naiveBayes(CP ~. , train)  
  
pred\_y\_tr = predict(model, train\_x)  
  
  
# Confusion Matrix  
cmt <- table(train$CP, pred\_y\_tr)  
# Model Evaluation  
confusionMatrix(cmt)

## Confusion Matrix and Statistics  
##   
## pred\_y\_tr  
## 0 1 2 3  
## 0 84 20 13 0  
## 1 5 22 9 2  
## 2 15 25 23 7  
## 3 8 3 3 5  
##   
## Overall Statistics  
##   
## Accuracy : 0.5492   
## 95% CI : (0.4844, 0.6127)  
## No Information Rate : 0.459   
## P-Value [Acc > NIR] : 0.0029191   
##   
## Kappa : 0.3314   
##   
## Mcnemar's Test P-Value : 0.0001818   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3  
## Sensitivity 0.7500 0.31429 0.47917 0.35714  
## Specificity 0.7500 0.90805 0.76020 0.93913  
## Pos Pred Value 0.7179 0.57895 0.32857 0.26316  
## Neg Pred Value 0.7795 0.76699 0.85632 0.96000  
## Prevalence 0.4590 0.28689 0.19672 0.05738  
## Detection Rate 0.3443 0.09016 0.09426 0.02049  
## Detection Prevalence 0.4795 0.15574 0.28689 0.07787  
## Balanced Accuracy 0.7500 0.61117 0.61969 0.64814

1. Validate the model on test data? Compute the actual versus predicted table, accuracy% and misclassification % on test data. Give your comments on model generalizability.

pred\_y = predict(model, test\_x)

## Warning in predict.naiveBayes(model, test\_x): Type mismatch between training  
## and new data for variable 'Result'. Did you use factors with numeric labels for  
## training, and numeric values for new data?

# Confusion Matrix  
cm <- table(test$CP, pred\_y)  
# Model Evaluation  
confusionMatrix(cm)

## Confusion Matrix and Statistics  
##   
## pred\_y  
## 0 1 2 3  
## 0 16 5 4 1  
## 1 0 8 3 1  
## 2 5 7 4 1  
## 3 1 1 2 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.4746   
## 95% CI : (0.343, 0.6088)  
## No Information Rate : 0.3729   
## P-Value [Acc > NIR] : 0.07076   
##   
## Kappa : 0.2455   
##   
## Mcnemar's Test P-Value : 0.31676   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3  
## Sensitivity 0.7273 0.3810 0.3077 0.00000  
## Specificity 0.7297 0.8947 0.7174 0.92857  
## Pos Pred Value 0.6154 0.6667 0.2353 0.00000  
## Neg Pred Value 0.8182 0.7234 0.7857 0.94545  
## Prevalence 0.3729 0.3559 0.2203 0.05085  
## Detection Rate 0.2712 0.1356 0.0678 0.00000  
## Detection Prevalence 0.4407 0.2034 0.2881 0.06780  
## Balanced Accuracy 0.7285 0.6378 0.5125 0.46429

1. Develop an optimum model to predict result using the Bagging method.

library(ipred)   
  
#fit the bagged model  
bag <- bagging(  
 formula = CP ~ .,  
 data = train,  
 nbagg = 75,   
 coob = TRUE,  
 control = rpart.control(minsplit = 2, cp = 0.069)  
)  
  
#display fitted bagged model  
bag

##   
## Bagging regression trees with 75 bootstrap replications   
##   
## Call: bagging.data.frame(formula = CP ~ ., data = train, nbagg = 75,   
## coob = TRUE, control = rpart.control(minsplit = 2, cp = 0.069))  
##   
## Out-of-bag estimate of root mean squared error: 0.9108

help(pack = ipred)

1. Display variable importance plot and give your comments

mymodel = randomForest(CP ~., data = train, mtry = 13, importance = TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer  
## unique values. Are you sure you want to do regression?

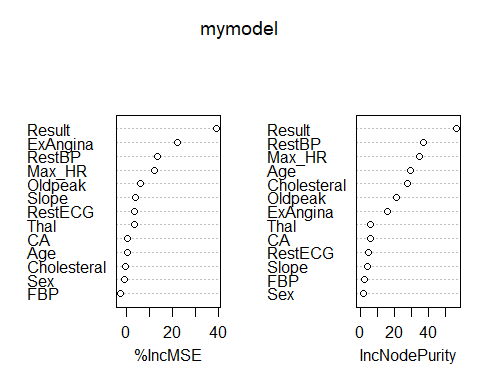
mymodel

##   
## Call:  
## randomForest(formula = CP ~ ., data = train, mtry = 13, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 13  
##   
## Mean of squared residuals: 0.8938037  
## % Var explained: 16.97

importance(mymodel)

## %IncMSE IncNodePurity  
## Age 0.3049806 29.676163  
## Sex -0.9004688 2.227207  
## RestBP 13.4855735 37.234712  
## Cholesteral -0.5741186 27.518024  
## FBP -2.9522036 2.813637  
## RestECG 3.3555424 4.741267  
## Max\_HR 11.9321115 34.820284  
## ExAngina 22.0535058 15.995078  
## Oldpeak 5.7362008 21.509038  
## Slope 3.6464034 4.170251  
## CA 0.3197481 6.105198  
## Thal 3.1457704 6.191542  
## Result 39.1690495 56.575561

varImpPlot(mymodel)



1. Compute the actual versus predicted table, accuracy% and misclassification % on training data. Give your comments on model accuracy.

pred\_y\_tr = predict(object = bag, newdata = train)  
  
u <- union(pred\_y\_tr, train$CP)  
  
t <- table(factor(pred\_y\_tr, u), factor(train$CP, u))  
  
confusionMatrix(t)

## Confusion Matrix and Statistics  
## Overall Statistics  
##   
## Accuracy : 0   
## 95% CI : (0, 0.015)  
## No Information Rate : 0.4795   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1.80090803527369 Class: 1.43025804521044  
## Sensitivity NA NA  
## Specificity 0.995902 0.98361  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.00000  
## Detection Rate 0.000000 0.00000  
## Detection Prevalence 0.004098 0.01639  
## Balanced Accuracy NA NA  
## Class: 1.36483854045195 Class: 1.32299360144752  
## Sensitivity NA NA  
## Specificity 0.991803 0.8443  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.0000  
## Detection Rate 0.000000 0.0000  
## Detection Prevalence 0.008197 0.1557  
## Balanced Accuracy NA NA  
## Class: 1.79115320812881 Class: 1.82292157103211  
## Sensitivity NA NA  
## Specificity 0.991803 0.9877  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.0000  
## Detection Rate 0.000000 0.0000  
## Detection Prevalence 0.008197 0.0123  
## Balanced Accuracy NA NA  
## Class: 1.3384486948773 Class: 1.36202448123645  
## Sensitivity NA NA  
## Specificity 0.97951 0.991803  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.000000  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.02049 0.008197  
## Balanced Accuracy NA NA  
## Class: 1.39141820314812 Class: 1.87550042594095  
## Sensitivity NA NA  
## Specificity 0.9877 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.0000 0.000000  
## Detection Rate 0.0000 0.000000  
## Detection Prevalence 0.0123 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.44571313864022 Class: 1.29106772978122  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.67908717903014 Class: 1.33673261429404  
## Sensitivity NA NA  
## Specificity 0.991803 0.8525  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.0000  
## Detection Rate 0.000000 0.0000  
## Detection Prevalence 0.008197 0.1475  
## Balanced Accuracy NA NA  
## Class: 1.28286596363144 Class: 1.34351723494384  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.77651484928043 Class: 1.37797758857469  
## Sensitivity NA NA  
## Specificity 0.97541 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.000000  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.02459 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.44399705805696 Class: 1.35218770772382  
## Sensitivity NA NA  
## Specificity 0.991803 0.9877  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.0000  
## Detection Rate 0.000000 0.0000  
## Detection Prevalence 0.008197 0.0123  
## Balanced Accuracy NA NA  
## Class: 1.25476003747353 Class: 1.2773287169347  
## Sensitivity NA NA  
## Specificity 0.98361 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.000000  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.01639 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.35109952760544 Class: 1.24102102462701  
## Sensitivity NA NA  
## Specificity 0.995902 0.97131  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.00000  
## Detection Rate 0.000000 0.00000  
## Detection Prevalence 0.004098 0.02869  
## Balanced Accuracy NA NA  
## Class: 1.3776791903016 Class: 1.31541130878592  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.90669651899413 Class: 1.76823598217803  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 1.29428893117092 Class: 0.496614839875271  
## Sensitivity NA NA  
## Specificity 0.995902 0.91803  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.00000  
## Detection Rate 0.000000 0.00000  
## Detection Prevalence 0.004098 0.08197  
## Balanced Accuracy NA NA  
## Class: 0.41185504696329 Class: 0.393494862347157  
## Sensitivity NA NA  
## Specificity 0.9877 0.877  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.0000 0.000  
## Detection Rate 0.0000 0.000  
## Detection Prevalence 0.0123 0.123  
## Balanced Accuracy NA NA  
## Class: 0.50968662133415 Class: 0.406566643806035  
## Sensitivity NA NA  
## Specificity 0.97541 0.93443  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.02459 0.06557  
## Balanced Accuracy NA NA  
## Class: 0.398783265504412 Class: 0.491326436718016  
## Sensitivity NA NA  
## Specificity 0.96721 0.94262  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.03279 0.05738  
## Balanced Accuracy NA NA  
## Class: 0.504398218176894 Class: 0.554274772535204  
## Sensitivity NA NA  
## Specificity 0.98361 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.000000  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.01639 0.004098  
## Balanced Accuracy NA NA  
## Class: 0.562300816981902 Class: 0.426746228467375  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 0.517508505784575 Class: 0.584822255862007  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 0.534095187873864 Class: 0.58940941843125  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 0.524577802838234 Class: 0.545279841673133  
## Sensitivity NA NA  
## Specificity 0.995902 0.995902  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.000000 0.000000  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.004098 0.004098  
## Balanced Accuracy NA NA  
## Class: 0.521023406414985 Class: 3 Class: 2 Class: 1  
## Sensitivity NA 0.00000 0.0000 0.0000  
## Specificity 0.995902 1.00000 1.0000 1.0000  
## Pos Pred Value NA NaN NaN NaN  
## Neg Pred Value NA 0.92213 0.7131 0.8443  
## Prevalence 0.000000 0.07787 0.2869 0.1557  
## Detection Rate 0.000000 0.00000 0.0000 0.0000  
## Detection Prevalence 0.004098 0.00000 0.0000 0.0000  
## Balanced Accuracy NA 0.50000 0.5000 0.5000  
## Class: 0  
## Sensitivity 0.0000  
## Specificity 1.0000  
## Pos Pred Value NaN  
## Neg Pred Value 0.5205  
## Prevalence 0.4795  
## Detection Rate 0.0000  
## Detection Prevalence 0.0000  
## Balanced Accuracy 0.5000

1. Validate the model on test data? Compute the actual versus predicted table, accuracy% and misclassification % on test data. Give your comments on model generalizability.

pred\_y= predict(object = bag, newdata = test)  
  
uu <- union(pred\_y, test$CP)  
  
tt <- table(factor(pred\_y, uu), factor(test$CP, uu))  
  
confusionMatrix(tt)

## Confusion Matrix and Statistics  
##   
##   
## Overall Statistics  
##   
## Accuracy : 0   
## 95% CI : (0, 0.0606)  
## No Information Rate : 0.4407   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1.24102102462701 Class: 1.35218770772382  
## Sensitivity NA NA  
## Specificity 0.98305 0.9661  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.0000  
## Detection Rate 0.00000 0.0000  
## Detection Prevalence 0.01695 0.0339  
## Balanced Accuracy NA NA  
## Class: 1.36655462103521 Class: 1.32299360144752  
## Sensitivity NA NA  
## Specificity 0.98305 0.8475  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.0000  
## Detection Rate 0.00000 0.0000  
## Detection Prevalence 0.01695 0.1525  
## Balanced Accuracy NA NA  
## Class: 1.77651484928043 Class: 1.81604374297782  
## Sensitivity NA NA  
## Specificity 0.91525 0.98305  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.08475 0.01695  
## Balanced Accuracy NA NA  
## Class: 1.31917365593913 Class: 1.3384486948773  
## Sensitivity NA NA  
## Specificity 0.98305 0.98305  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.01695 0.01695  
## Balanced Accuracy NA NA  
## Class: 1.33673261429404 Class: 1.37626150799143  
## Sensitivity NA NA  
## Specificity 0.8814 0.98305  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.0000 0.00000  
## Detection Rate 0.0000 0.00000  
## Detection Prevalence 0.1186 0.01695  
## Balanced Accuracy NA NA  
## Class: 1.44399705805696 Class: 1.76823598217803  
## Sensitivity NA NA  
## Specificity 0.98305 0.98305  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.01695 0.01695  
## Balanced Accuracy NA NA  
## Class: 1.87550042594095 Class: 0.393494862347157  
## Sensitivity NA NA  
## Specificity 0.98305 0.8475  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.0000  
## Detection Rate 0.00000 0.0000  
## Detection Prevalence 0.01695 0.1525  
## Balanced Accuracy NA NA  
## Class: 0.496614839875271 Class: 0.50968662133415  
## Sensitivity NA NA  
## Specificity 0.9322 0.9322  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.0000 0.0000  
## Detection Rate 0.0000 0.0000  
## Detection Prevalence 0.0678 0.0678  
## Balanced Accuracy NA NA  
## Class: 0.406566643806035 Class: 0.41185504696329  
## Sensitivity NA NA  
## Specificity 0.98305 0.94915  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.01695 0.05085  
## Balanced Accuracy NA NA  
## Class: 0.54019401397269 Class: 0.512220102627319  
## Sensitivity NA NA  
## Specificity 0.98305 0.98305  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.00000  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.01695 0.01695  
## Balanced Accuracy NA NA  
## Class: 0.521023406414985 Class: 0.398783265504412  
## Sensitivity NA NA  
## Specificity 0.98305 0.9661  
## Pos Pred Value NA NA  
## Neg Pred Value NA NA  
## Prevalence 0.00000 0.0000  
## Detection Rate 0.00000 0.0000  
## Detection Prevalence 0.01695 0.0339  
## Balanced Accuracy NA NA  
## Class: 0.504398218176894 Class: 0 Class: 1 Class: 3  
## Sensitivity NA 0.0000 0.0000 0.0000  
## Specificity 0.98305 1.0000 1.0000 1.0000  
## Pos Pred Value NA NaN NaN NaN  
## Neg Pred Value NA 0.5593 0.7966 0.9322  
## Prevalence 0.00000 0.4407 0.2034 0.0678  
## Detection Rate 0.00000 0.0000 0.0000 0.0000  
## Detection Prevalence 0.01695 0.0000 0.0000 0.0000  
## Balanced Accuracy NA 0.5000 0.5000 0.5000  
## Class: 2  
## Sensitivity 0.0000  
## Specificity 1.0000  
## Pos Pred Value NaN  
## Neg Pred Value 0.7119  
## Prevalence 0.2881  
## Detection Rate 0.0000  
## Detection Prevalence 0.0000  
## Balanced Accuracy 0.5000

1. Develop a model to predict result using the Random Forest method.

mymodel = randomForest(CP ~., data = train, importance = TRUE)  
mymodel

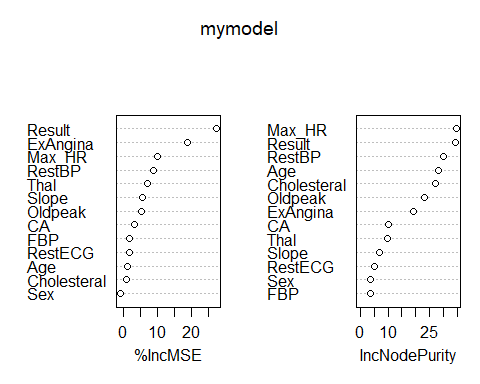
##   
## Call:  
## randomForest(formula = CP ~ ., data = train, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## Mean of squared residuals: 0.8181326  
## % Var explained: 24

1. Display variable importance plot and give your comments

importance(mymodel)

## %IncMSE IncNodePurity  
## Age 1.1834613 28.170212  
## Sex -0.8939923 3.599703  
## RestBP 8.7912906 30.006374  
## Cholesteral 0.9630746 26.865687  
## FBP 1.9368725 3.592575  
## RestECG 1.8069184 5.138652  
## Max\_HR 10.1354431 34.670196  
## ExAngina 18.7043267 19.127504  
## Oldpeak 5.2991589 23.032267  
## Slope 5.5418174 7.117615  
## CA 3.2771599 10.103610  
## Thal 7.1393276 9.820835  
## Result 27.3293142 34.025461

varImpPlot(mymodel)



1. Compute the actual versus predicted table, accuracy % and misclassification % on training data. Give your comments on model accuracy.

pred\_y\_tr = predict(mymodel, train)  
  
u <- union(pred\_y\_tr, train$CP)  
  
t <- table(factor(pred\_y\_tr, u), factor(train$CP, u))  
  
confusionMatrix(t)

1. Validate the model on test data? Compute the actual versus predicted table, accuracy % and misclassification % on test data. Give your comments on model generalizability.

pred\_y= predict(object = mymodel, newdata = test)  
  
uu <- union(pred\_y, test$CP)  
  
tt <- table(factor(pred\_y, uu), factor(test$CP, uu))  
  
confusionMatrix(tt)

1. Compare the classification tree, bagging & random forest models and give your comments.

From Regression tree, bgging & random forest models we can easily compute that the model using Random Forest is best most model as comparision because the value of specificity of Random Forest Model is high training and testing data respectively.

1. Compare the classification tree, bgging & random forest models with the logistic regression model of assignment 2 and give your comments.